ProDA: An End-to-End Wavelet-Based OLAP System for Massive Datasets

Cyrus Shahabi, Mehrdad Jahangiri, and Farnoush Banaei-Kashani
University of Southern California

ProDA employs wavelets to support exact, approximate, and progressive OLAP queries on large multidimensional datasets, while keeping update costs relatively low. ProDA not only supports online execution of ad hoc analytical queries on massive datasets, but also extends the set of supported analytical queries to include the entire family of polynomial aggregate queries as well as the new class of plot queries.

Recent advancements in sensing and data acquisition technologies have enabled collection of massive datasets that represent complex real-world events and entities in fine detail. In light of access to such datasets, scientists and system analysts are no longer restricted to modeling and simulation when analyzing real-world events. Instead, the preferred viable approach derives observations and verifies hypotheses by analytical exploration of representative real datasets that capture the corresponding event. This approach demands intelligent data storage, access, and analytical querying solutions and tools that facilitate convenient, efficient, and effective exploration of these massive datasets. This poses both an opportunity and a Grand Challenge for the database community.[1]

By design, developers optimize traditional databases for transactional rather than analytical query processing. These databases support only a few basic analytical queries with nonoptimal performance and, therefore, provide inappropriate tools for analyzing massive datasets.

Instead, current practice exploits the extensive analytical query processing capabilities of spreadsheet applications such as Microsoft Excel and Lotus 1-2-3 to explore the data. However, in this case the limitation lies in the spreadsheet applications' capability to handle large datasets. With this approach, while the original datasets still reside in a database server, smaller subsets of the data will be selected—by sampling, aggregation, or categorization—and retrieved as new data products for further local processing with the spreadsheet application at the client side.

This inconvenient and time-consuming process of generating a secondhand dataset might unavoidably result in loss of relevant or detailed information. This can bias the analysis and encourage analysts to justify their own assumptions rather than discover surprising latent facts from the data. Further, given that most of the analysis occurs locally at the client with this approach, the data transfer overhead is increased, resource sharing at the server side does not apply, and considerable processing is required at the client side.

Online analytical processing tools have emerged to address the limitations of traditional databases and spreadsheet applications. Unlike traditional databases, OLAP tools support a range of complex analytical queries; unlike spreadsheet applications, they can also handle massive datasets. Moreover, OLAP tools can process user queries online. Online query processing is arguably

0018-9162/08/$25.00 © 2008 IEEE Published by the IEEE Computer Society

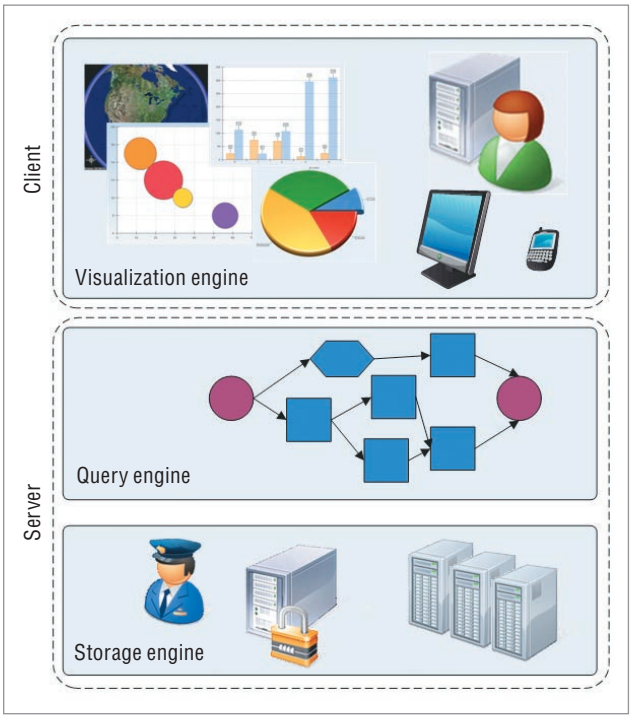

Figure 1. ProDA architecture. The system supports a wide range of analytical queries while being able to handle massive datasets. The storage tier maintains the data while the query tier executes the queries. Together, these elements comprise the ProDA server. The ProDA client, on the other hand, implements the visualization tier on top, where user queries are formulated and results are presented.

a requirement for effectively supporting exploratory data querying. However, current OLAP tools heavily rely on precalculating the query results to enable online query processing. Consequently, they can support only a limited set of predefined (rather than ad hoc) queries online.

Over the past half decade, we have designed, developed, and matured an end-to-end system, dubbed ProDA (for progressive data analysis system), that efficiently and effectively analyzes massive datasets. ProDA functions as a client-server system with the three-tier architecture shown in Figure 1: The storage tier maintains the data at the bottom while the query tier executes the queries at the midlevel; together these elements comprise the ProDA server. The ProDA client, on the other hand, implements the visualization tier on top, where user queries are formulated and query results are presented.

As an OLAP tool, ProDA supports a wide range of analytical queries while also being able to handle massive datasets. However, compared to current OLAP tools, ProDA offers the extended and enhanced online query processing capabilities made possible by leveraging our in-house wavelet-based technology.[2,3]

Specifically, ProDA supports more complex analytical queries, including the entire family of polynomial aggregate queries as well as the class of plot queries previously unsupported by OLAP tools. Moreover, unlike current OLAP tools, ProDA supports online ad hoc queries. To enable online execution of these queries, we take two measures to improve the efficiency of query execution. First, we treat analytical queries as database queries and push them down, close to the data, to be executed at the server side rather than in client-side applications. Second, we leverage the wavelet transform's excellent energy-compaction properties, which allow for accurate approximation of the query result with minimal data access. Here, we innovate by transforming the query as well as the data.

Since queries are often more patterned than data, they are also more compactable when transformed into the wavelet domain. With a highly compact yet accurate query representation, in addition to a compact data representation, we can effectively select and retrieve the high-energy data coefficients relevant to the query with exponentially less data access when compared to previous approaches that only transform data. Therefore, we can approximate the query result accurately with an exponentially improved response time. In combination, these two measures let ProDA carry out the online execution of ad hoc queries.

Further, leveraging the multiresolution properties of the wavelet transform can answer approximate queries progressively, either with fixed accuracy or, alternatively, with fixed performance such as a limited time frame. This feature is particularly useful for exploratory data analysis, which can be quite time- and resource-intensive with massive datasets.

With exploratory analysis, users often issue several back-to-back queries, each time revising and enhancing a query based on quick observation of the previous query's partial results. With progressive queries, users can save time and system resources according to the required query accuracy or available time and system resources to execute each query.

On the other hand, with ProDA—inline with the typical use of wavelet transform in databases—we can optionally drop the transformed data's low-energy coefficients to save storage space. Since the storage space is no longer the main resource constraint, we prefer lossless data storage to allow for exact query answering, if needed.

Finally, ProDA also introduces novel operators that let developers manipulate the stored data by inserting, deleting, and updating data records directly in the wavelet domain rather than in the original domain. These operators are extremely useful for maintaining the massive datasets, particularly when the data is frequently updated by, for example, incoming data streams; otherwise, the entire dataset must be transformed back to the original domain for any minor data manipulation.

CASE STUDY

We have successfully used ProDA to analyze real-world data applications. These case studies, such as the following oilfield sensor data analysis study, served as proofs of concept and as testbeds for realizing and addressing ProDA's practical limitations.

With recent advancements in sensor technology, we can now economically equip oil production wells and an oilfield's water and steam injection wells with multimodal sensor devices to monitor gas, water, steam, oil pressure, and related factors. Smart oilfield management systems (http://cisoft.usc.edu, for example) must analyze such a data feed in real time to provide decision support for the oilfield operators. The system can also extend to automatically control the oilfield when it creates a closed loop.

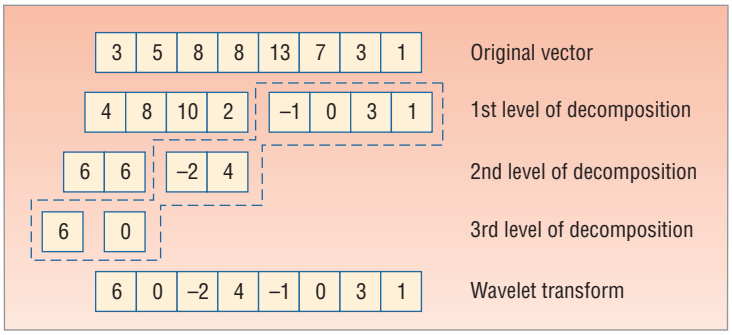

Figure 2. Wavelet transform. To perform this transform, we first compute the pairwise average and pairwise difference for all consecutive pairs of data values in the original vector. The result consists of two vectors, each of half size, including a smooth average view and a rough difference view. We repeat this process.

Analyzing oilfield sensor data is complicated not only by the data's size and many dimensions, but also because of the data's high update rate, which renders any slow analysis process useless. With this application, domain experts must execute complex ad hoc queries on the fly to understand the oilfield's dynamic behavior in real time and react accordingly.

To illustrate, imagine that the oilfield management system is continually receiving gas, water, steam, and oil pressure readings from a field of 4,000 wells, where each well has 20 sensor devices with a sampling period of 15 seconds—deployed at the well's various depths. Typically, a reservoir engineer must continuously monitor the covariance between the water and steam injection and oil production across all wells. The covariance matrix determines the injection rates required for optimal total production. As far as we know, only ProDA can compute such a complex query on the fly.

UNDERLYING TECHNOLOGY

The technology ProDA uses to manage massive data relies heavily on tools such as wavelets.

Preliminaries: Wavelet transform

Developers originally adopted the wavelet transform tool from the signal processing literature for multiscale decomposition of data signals. Wavelets can transform a data signal into a pair of rough and smooth views. The rough view captures the data's low-energy components, whereas the smooth view represents the high-energy components. The tool then iteratively applies the transformation to each view to generate more pairs of rough and smooth views at lower resolutions. Eventually, the transformed data appears as a combination of selected views at different resolutions.

Wavelet transform is reversible. Thus, the system can use the transformed data to reconstruct the original data at any resolution, including the original one. With wavelet transform, the system generates the smooth and rough views by applying low-pass and high-pass filters to the data, respectively. In the case of discrete wavelet transform (DWT), a filter simply consists of a coefficient pair. For example, in the case of the basic Haar wavelet transform, the low-pass filter is

$$\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$$

and the high-pass filter is

$$\left(\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right).$$

To apply, the filter is convoluted with the signal to generate the transformed view. The following example simplifies the demonstration by using (½, ½) and (½, –½) as filters.

Data reconstruction

Consider the vector of 8 values (3, 5, 8, 8, 13, 7, 3, 1) depicted in Figure 2. To perform wavelet transform, we first compute the pairwise average and pairwise difference for all consecutive pairs of data values in the original vector. The result consists of two half-size vectors, including a smooth average view (4, 8, 10, 2) and a rough difference view (–1, 0, 3, 1) that, together, form the data's first-level decomposition.

Next, we continue by similarly constructing the average and difference views for the average view from the first-level decomposition. The next-lower resolution average and difference views are (6, 6) and (–2, 4), respectively. Finally, by repeating the process, we find the average view, (6), and the difference view, (0), for the average view from the second-level decomposition. These views form the decomposition's third and final level. By combining the final average and the differences produced at all levels of decomposition,

we form the original vector's wavelet transform: (6, 0, –2, 4, –1, 0, 3, 1).

Wavelet-based online query processing

In the database literature, developers often use wavelet transform for data compression and not for efficient query execution as we do with ProDA. This approach transforms the data and, to save storage space, it preserves only the transformed data's high-energy coefficients. This is by nature a lossy transformation and, as an unwanted consequence, querying such data inevitably results in approximate results. Moreover, the data compression's efficacy is highly data-dependent, and compression is only effective when the data has a concise wavelet approximation.

Instead, we use wavelets to approximate the incoming queries rather than the data. If certain conditions are met, the queries are well compressible, and query execution in the wavelet domain thus functions efficiently. With ProDA, although the dataset is still wavelet transformed, it is not necessarily approximated because we can optionally maintain all coefficients of the transformed data. As a result, ProDA can execute queries accurately and efficiently, independent of the data characteristics.

Data compression is essential for real-world situations that deal with either extremely large data or limited storage capability.

On the other hand, the query's desired accuracy varies per application, user, or dataset, and therefore offers an opportunity to trade off accuracy for improved response time. ProDA lets progressive query execution exploit this opportunity. Intuitively, wavelet transform preserves the data's energy but redistributes it across all wavelet coefficients such that most of the energy is localized in a few coefficients. These high-energy wavelet coefficients are more significant than others in representing the data. By ordering and progressively retrieving the coefficients according to their significance, ProDA can quickly produce an accurate estimation of the query result, which progressively converges to the exact result.

BOTTOM TIER: STORAGE ENGINE

The bottom tier of ProDA—the storage engine—stores and manages the data. This engine consists of several different data sources: user accounts, a history of user activities, a datacube directory, and datacubes. Each datacube consists of

- general information about the cube,
- a list of user-defined polynomial queries,
- the dimension values of datacubes, and
- the wavelet-transformed measure values.

With the exception of the Wavelet DB, we have implemented all the data sources in a relational database server to facilitate the management of the data. For the Wavelet DB, we have developed our own custom-made binary file structures to store multidimensional wavelet-transformed data using our optimal disk block allocation strategy in order to achieve high efficiency.

Disk block allocation

In practice, a disk block can store more than one data coefficient. This presents the challenge of optimizing the placement of the wavelet coefficients in the secondary storage (for example, magnetic disk drives), such that the number of disk blocks retrieved to answer queries is minimized. In other words, we must lay the data out on the disk in such a way that the *principle of locality of reference* holds, that is, when a datum is accessed on a disk block, it is likely for other data on the same block to be accessed simultaneously.

We have observed that OLAP queries on wavelet data require a distinct access pattern, such that when the system retrieves a wavelet coefficient, it guarantees that all its dependent coefficients will also be retrieved. Hence, we have exploited this unique access pattern and designed a disk-placement strategy for wavelet data that yields the best possible I/O complexity for query evaluation.[4]

Query-aware data compression

Many real-world situations deal with either extremely large data or limited storage capability. For either of these, data compression is essential. With ProDA to compress the data (assuming that the storage space is limited), we can store only the most significant wavelet coefficients. The most significant coefficients are those with either the lowest frequencies or the highest energy. We define the energy of a coefficient as the power two of the coefficient value. We take this approach because high-frequency coefficients or small coefficients can be considered as noise and, thus, could be dropped.

Next, if a query requests a data coefficient that is not stored, such as one dropped previously due to data compression, we assume the coefficient equals zero and continue the process. This assumption basically implements *hard thresholding*. Using such a compression schema, we provide excellent approximate query results with ProDA while maintaining the query response time independent of the compression ratio.

MIDDLE TIER: QUERY ENGINE

The query engine of ProDA provides a rich set of Web services that consists of four groups: browsing services,

essential querying services, advanced querying services, and data mining services. Figure 3 illustrates these categories in four layers, as each is built atop another.

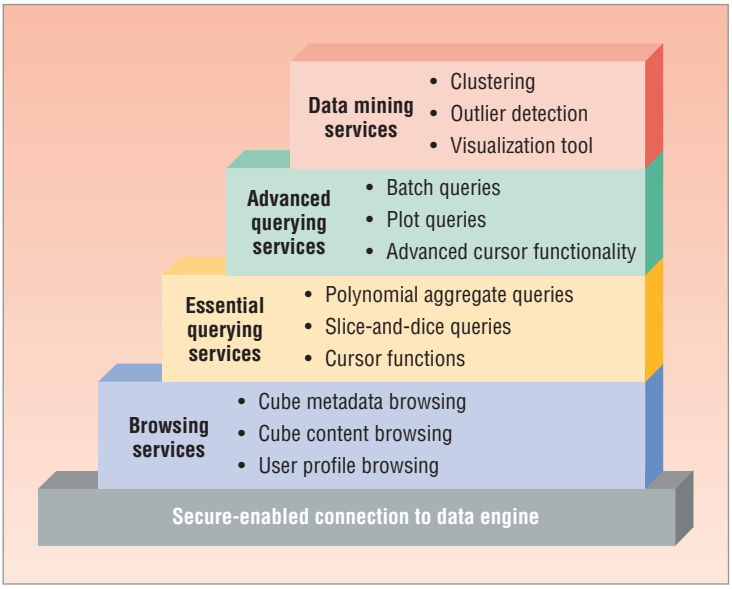

Figure 3. ProDA's query engine. The engine provides a rich set of Web services consisting of four groups of services.

Browsing services

We have designed this group of Web services to allow users to manage, explore, and modify the available datacubes.

Cube metadata browsing. These services let users explore the metadata of the available datacubes, such as the description, schema, and the wavelet filter used for datacube transformation. Users can add, modify, or drop the user-defined queries, and can browse through and reuse previously issued queries.

Cube content browsing. This family of services lets users directly access the content of the selected datacube. Using this set, we can add or drop a datacube, or modify a datacube's wavelet coefficients by issuing update queries.

User profile browsing. This family of services is used to implement the user access control. Besides, the services provide facilities for the users to add, drop, and modify the user profile information.

Essential querying services

This class of services includes polynomial aggregate queries, slice-and-dice queries, and cursor functions.

Polynomial aggregate queries. The standard statistical range-aggregate queries are implemented as predefined queries in ProDA—for example, count, sum, average, variance, covariance, and correlation. Furthermore, we have broadened the supported queries to encompass ad hoc statistical functions by providing a formal method of expressing analytical queries as polynomial functions on the data attributes. Users can define any polynomial expression on attributes and share the definition with others. For example, a high-ordered polynomial function such as *kurtosis* (the fourth moment of data divided by the square of the variance) is not predefined in ProDA; however, our new utility environment lets users define and share such complex queries on-the-fly.

We implement an arbitrary polynomial query by using two basic Web services, *PushTerm* and *PushOperator*. Given the importance of the order of the function calls, ProDA parses the equations in post order. For instance, consider implementation of the variance function using PushTerm and PushOperator. Variance is defined as follows:

$$Var(x) = \frac{\sum x_i^2}{n} - \left(\frac{\sum x_i}{n}\right)^2$$

The post order representation of this function is $\sum x_i^2, n, /, \sum x_i, n, /, ^2$, and $-$. Accordingly, we implement our polynomial query by executing the following [9] calls:

```
PushTerm(x, 2); PushTerm(x, 0);
  PushOperator(´/´); PushTerm(x, 1);
  PushTerm(x, 0);
PushOperator(´/´); PushOperator(´**´);
  PushOperator(´—´); Submit();
```

Slice-and-dice queries. Consider the scenario in which we wish to extract a region of the original data from its wavelet transform. This poses the following dilemma: We can either reconstruct the entire dataset and extract the desired region (which is infeasible), or reconstruct the desired region point-by-point (which is inefficient, particularly for large regions). Instead, we translate the selected operation of the relational algebra to the wavelet domain and choose the required coefficients for reconstruction of the desired range.[5] By employing this technique, ProDA clients can access small subsets of the wavelet data instantly because the server never needs to reconstruct the entire dataset. This class of queries is used when ProDA users intend to download a small subset of data into their own machines for convenient interaction. For example, an oil production engineer usually analyzes a few production oil wells at a time, without accessing the data of the entire reservoir. This relevant data is usually small enough to be cached in the client machine. The cached data is readily updatable and enables efficient query processing. Meanwhile, ProDA allows the user to receive the exact result when the remote connection is available.

```
// Creating an instance and storing session state
ProDAWebServices pws=new ProDAWebServices();
// Selecting a cube with the login information
pws.SelectDB(dbName,userName,password);
// Defining a range and submitting a query
pws.SetRange(lowerLeft,upperRight);pws.Variance(1);
// Asking for result progressively
while(pws.HasMore())
    Console.Write("Result="+pws.Advance(5%));
```

Figure 4. Sample ProDA client for progressive querying in C#. The client uses cursor services to obtain the query result progressively.

Cursor functions. ProDA provides progressive query answering by ordering the wavelet coefficients based on the significance of the query coefficients. Hence, ProDA incrementally retrieves the data coefficients related to each query from the storage engine. Cursor functions track the progress of the query operations and the data retrieval operations. They also let users stop the query processing any time they are satisfied with the intermediate approximate result.

Advanced querying services

Utilizing the essential querying services, we efficiently implement two widely used advanced queries—batch and plot queries. Furthermore, we provide additional cursor functionality for these two query classes by prioritizing certain query regions.

Batch queries. Scientists typically submit queries in batch rather than issuing individual, unrelated queries. We have proposed a wavelet-based technique that exploits I/O sharing across a query batch for efficient and progressive evaluation of the batch queries. The challenge is that controlling the structure of errors across the query results now becomes more critical than minimizing errors per each individual query. We have defined a class of structural error-penalty functions in our framework to achieve the optimal progressiveness for a batch queries.[6]

Users can invoke the batch query services by specifying a grid over data dimensions. Thereafter, they can progressively receive the results for the entire batch.

Plot queries. Plots are among the most important and widely used tools in scientific data analysis and visualization applications. In general, each plot point is an aggregate value over one or more measure attributes for a given dimension value. The current practice for generating a plot over a multidimensional dataset involves computing the plot point-by-point, where each point is the result of computing an aggregate query. Therefore, for large plots a large number of aggregate queries are submitted to the database. This method is neither memory-efficient (on either the client or the server side) nor communication-efficient.

On the other hand, we redefine a plot as a single database query and employ a wavelet-based technique that exploits I/O sharing across the aggregate queries for all plot points to evaluate the plot efficiently. With this approach, the main idea is to decompose a plot query into two sets: a set of aggregate queries and a set of slice-and-dice queries. Subsequently, we can use our earlier results to compute both sets of queries efficiently in the wavelet domain.

Users can invoke the plot query services by specifying a range over the data dimensions and selecting the independent variable for the plot. A developer can employ the following advanced cursor functionality to generate the plot output progressively.

Advanced cursor functionality. Scientists consider batch and plot queries among the most favored statistical analysis tools. They are widely used to provide valuable insights about any dataset. For example, we can extract outliers, trends, clusters, or local maxima by quickly looking at the output of such queries. Furthermore, the entire query result often is not used at once. For instance, the result might not fit on the screen, the user might point to a specific region of the result, or the user might prioritize the subsets of the result—for example, local maxima, or the regions with high values or high gradients, to be computed. Accordingly, the advanced cursor functionality allows users to modify the structural-error-penalty functions to control the progressiveness of the query.

Data mining services

We are currently designing and implementing additional analytical query processing components in ProDA to support complex data mining functionalities. So far, we have enhanced ProDA by incorporating clustering and outlier detection techniques, and an effective visualization tool for exploratory data mining. For clustering, we use various methods—such as K-means with different distant functions—to cluster batch or plot query results. Using a similar approach with a customized penalty function, ProDA enables the progressive answering of the outlier detection queries. The visualization tool generates time-stamped KML files to be imported to Google Earth for effective spatial and temporal visualization.

TOP TIER: VISUALIZATION

By pushing the extensive and complex data processing tasks to the server side, the ProDA client can be implemented as a light and yet effective interface.

Figure 4 demonstrates a sample client that invokes ProDA Web services for query processing. First, we create an instance of ProDA services, then select a datacube with appropriate login information. Next, we specify a range and submit a variance query. Finally, we use our cursor services to obtain the query result progressively.

In addition, we have developed a stand-alone graphical C# client, ProDA client, for efficient interaction with arbitrary scientific datasets. We emphasize using the graphical interface as a more intuitive query interface. We have incorporated smart client functionalities (for example, smart data management, online/offline capability, high-fidelity UI) into ProDA client; thus, ProDA provides an adaptive, responsive, and rich interactive experience by leveraging local resources and intelligently connecting to distributed data sources.

The ProDA client consists of data and query visualization, high-fidelity UI, connectivity management, and advanced visualization. In short, ProDA lets the user select a datacube and visualize the data. It also accepts queries from a user and displays the results progressively in offline or online mode, provides resource sharing at the client machine, and utilizes advanced commercial visualization tools.

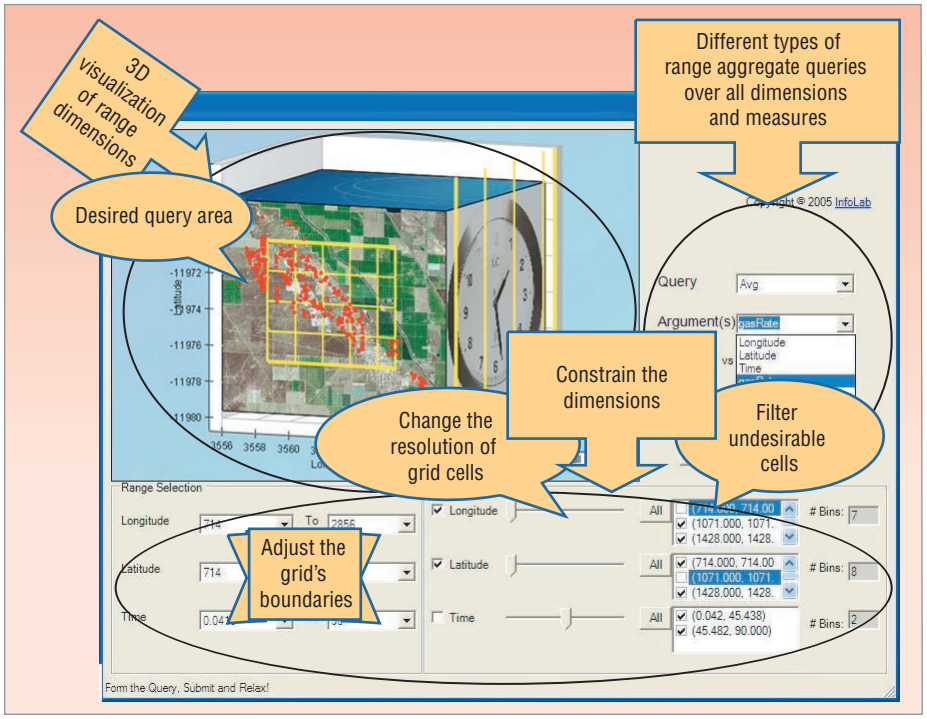

Figure 5. Data visualization. This sample data visualization shows a particular dataset with a user-specified grid for batch query processing. Data visualization modules exhibit various attribute types, including spatial, temporal, numeric, and categoric.

Data visualization

This module lets the user log in to the ProDA storage engine, browse the data sources, and select one to interact with. It then provides a visualization of the selected dataset for the user to browse, scroll, and rotate. It also facilitates definition of the desired ranges and queries. Defining a bounding box for a single range query and a grid for batch queries over all dimensions is one of many necessary functionalities the data visualization module must provide. Figure 5 demonstrates a sample oilfield sensor data visualization. The grid is specified by the user for batch query processing.

Data visualization modules exhibit various attribute types, including spatial, temporal, numeric, and categoric. In addition to presenting the hierarchy of the dimension values, ProDA displays dependent dimensions and allows the user to work with all dimensions simultaneously. For example, while exploring the oilfield sensor data, we can select a set of oil wells by identifying a certain window of interest or, alternatively, by choosing the wells based on their corresponding labels.

Once a developer selects a datacube, ProDA enables the list of available queries. This list includes common analytical queries, user-defined polynomial range-aggregate queries, plot queries, and slice-and-dice queries. In addition, the user can define a new polynomial query, add it to the list, and even share it with other users of this dataset.

Query visualization

The ProDA client displays the output of queries once they become available, then updates them frequently as new updates appear. It also visualizes the output of batch queries and plot queries using various advanced built-in chart types. As a query progresses, the user can start interacting with the result through the ProDA visualization module's operations. Zooming in and out, pivoting, and exporting are among the many possible actions.

When real-world data contains noise, the output of a plot query displays some undesired small variations. These variations not only do not carry any valuable information, but also confuse the users when analyzing the data. Leveraging from wavelet hard threshholding, ProDA supports advanced denoising of the query output and generates smoother outputs, especially when they carry white Gaussian noise.

High-fidelity UI

ProDA, as a well-designed smart client, guarantees that the utilities already installed on the client machine can access and process the data that ProDA generates. This is essential in practice because typical users are accustomed to using Microsoft Office components,

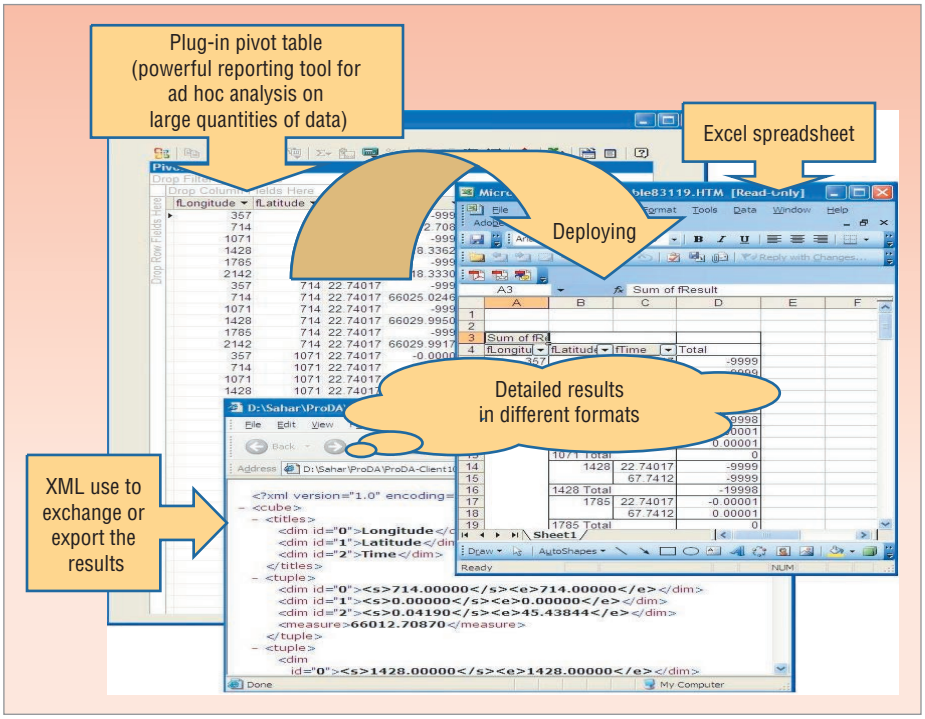

Figure 6. ProDA's export capability. With its extensive set of export functionalities, the system can be connected to almost any application. At any time, users can export its data to XML, Excel, text files, and many more formats.

especially Microsoft Excel, in addition to ProDA's built-in visualization packages.

Toward this end, the ProDA client provides a similar interface to improve the users' performance and decrease training costs. In particular, we use the Microsoft Excel pivot table components designed for ad hoc analysis of large quantities of data. A pivot table is a powerful reporting tool that features basic to complicated calculation modules independent of the spreadsheet's original data layout. With the table's drag-and-drop function, users can pivot the data and perform local computation tasks. Consequently, users will have an interactive table that automatically extracts, organizes, and summarizes the data. They can use this report to analyze the data, make comparisons, detect patterns and relationships, and discover trends.

With its extensive set of export functionalities, ProDA can be connected to almost any application. At any time, users can export the data to XML, Excel, text files, and many more formats. Figure 6 shows ProDA's export functionality.

Connectivity management

ProDA client lets a user cache a subset of a dataset while the system is online. Later, when the system is offline, by utilizing the cached data, the user can access all ProDA's functionalities as if the system were still connected to the server. Offline query processing offers an especially attractive feature for mobile users who need access to the data during a disconnected operation. Hence, we will empower ProDA clients to support offline wavelet-based query processing. With ProDA, the user can query the wavelet-transformed sketch to receive an excellent approximate answer.

Advanced visualization

We employ other widely accepted commercial products for universal spatial data representation. In particular, ProDA exports the spatial query results to Google Earth for advanced visualization. This tool allows a group of users to exchange their query results with each other while viewing other spatial data in relation to the problem.

ProDA enables exploratory analysis of massive multidimensional datasets. Standard OLAP systems that rely on query precalculation are expensive to update, whereas traditional, easily updatable databases often have poor response time with analytical queries. With ProDA, we employed wavelets to support exact, approximate, and progressive OLAP queries on large multidimensional datasets, while keeping update costs relatively low. ProDA extends the set of supported analytical queries to include the entire family of polynomial aggregate queries as well as the new class of plot queries.

Acknowledgments

This research has been funded in part by NSF grants EEC-9529152 (IMSC ERC) and IIS-0238560 (PECASE), unrestricted cash gifts from Google and Microsoft, and by NASA's JPL SURP program and the Center for Interactive Smart Oilfield Technologies (CiSoft). CiSoft is a joint University of Southern California/Chevron initiative.

References

1. M. Stonebraker et al., "The Lowell Database Research Self-Assessment," *Comm. ACM*, vol. 48, no. 5, 2005, pp. 111-118.
2. M. Jahangiri and C. Shahabi, *Wolap: Wavelet-Based Range Aggregate Query Processing*, tech. report, Dept. Computer Science, Univ. of Southern California, 2007.

3. R. Schmidt and C. Shahabi, "Propolyne: A Fast Wavelet-Based Technique for Progressive Evaluation of Polynomial Range-Sum Queries," *Proc. Extending Database Technology Conf.* (EDBT 02), Springer, 2002, pp. 664-681.
4. C. Shahabi and R. Schmidt, *Wavelet Disk Placement for Efficient Querying of Large Multidimensional Data Sets*, tech. report, Dept. of Computer Science, Univ. of Southern California, 2004.
5. M. Jahangiri, D. Sacharidis, and C. Shahabi, "SHIFT-SPLIT: I/O Efficient Maintenance of Wavelet-Transformed Multidimensional Data," *Proc. ACM SIGMOD*, ACM Press, 2005, pp. 275-286.
6. R. Schmidt and C. Shahabi, "How to Evaluate Multiple Range-Sum Queries Progressively," *Proc. ACM PODS*, ACM Press, 2002, pp. 133-141.

Cyrus Shahabi *is an associate professor and the director of the Information Laboratory (InfoLAB) at the University of Southern California's Computer Science Department. His research interests include geospatial and multidimensional data analysis, peer-to-peer systems, and streaming architecture. Shahabi received a PhD in computer science from the University of Southern California. Contact him at cshahabi@cs.usc.edu.*

Mehrdad Jahangiri *is a PhD candidate in the Department of Computer Science at the University of Southern California. His research interests include OLAP, data integration, and data mining. He received an MS in computer science from the University of Southern California. Contact him at jahangir@usc.edu.*

Farnoush Banaei-Kashani *is a postdoctoral research associate in the Department of Computer Science at the University of Southern California. His research interests include data-stream systems, networked databases, and group data management. He received a PhD in computer science from the University of Southern California. Contact him at banaeika@usc.edu.*